




















Хичээл 1: Shell scripting

Г.Гантулга

2024 оны 11-р сарын 24

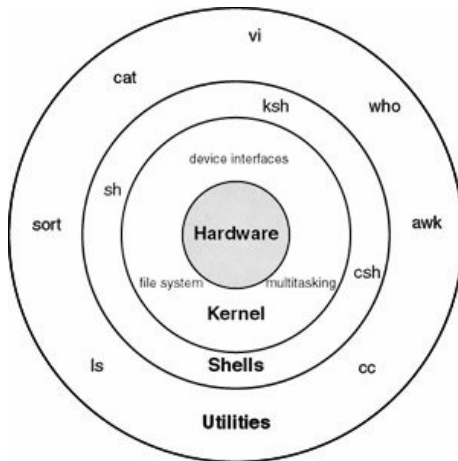
- 1 Shell скриптийн танилцуулга
- 2 Өргөн ашиглагдах командууд
- 3 Shell скрипт

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE
EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	 4 WEEKS	 3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	 8 WEEKS	 6 DAYS	 1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	 4 WEEKS	 6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	 5 WEEKS	 5 DAYS	 1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	 10 DAYS	 2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	 2 WEEKS	 1 DAY
	 1 DAY					 8 WEEKS	 5 DAYS

Зүгээр: <https://xkcd.com/1205/>

Shell гэж юу вэ?



Shell-ийн үүрэг

- 1 Хэрэглэгчээс командыг уншина
- 2 Командыг өөрөөсөө `fork()` хийн ажиллуулна. Нэг эсвэл `background`-аар эсвэл `foreground`-д ажиллуулна.
 - ▶ **background**: *команд &*
 - ▶ **foreground**: *команд*
- 3 Ажиллаж буй процессын оролт, гаралтыг хариуцна.

shell дотор хэрэгжүүлсэн командууд

- `jobs`
- `fg`
- `bg`
- `cd`

Хэрэглэгчтэй харилцах shell prompt

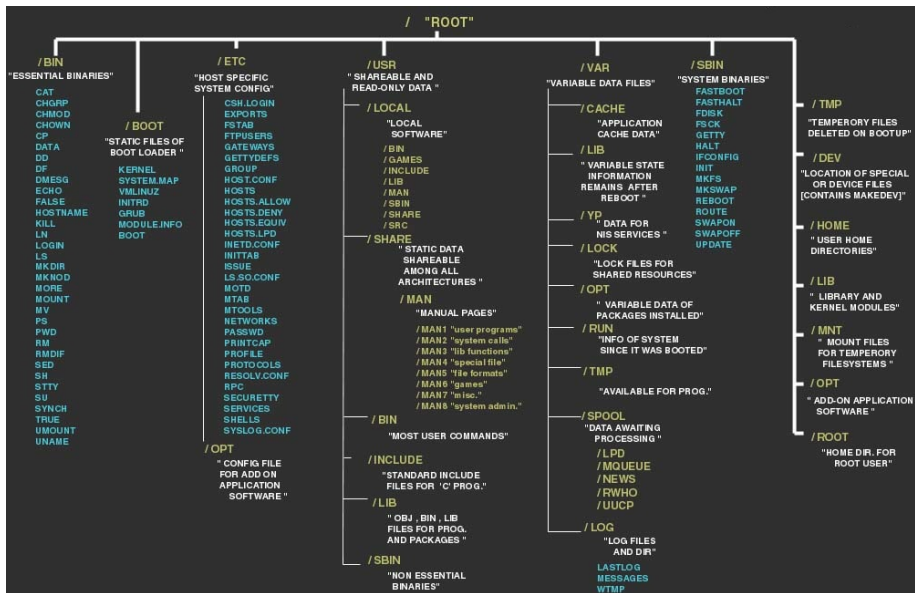
```
ggt@joy:~$
```

ROOT файл систем болон PATH хувьсагч

```
$ echo $PATH
```

```
/home/ggt/.local/bin:/usr/local/sbin:/usr/local/bin:  
/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/  
games:/snap/bin:/snap/bin
```

ROOT файл систем болон PATH хувьсагч



Өргөн ашиглагдах командууд

Текст гаралт, текст файлдай ажиллах командууд

- echo
- more / less
- head / tail
- cat
- grep
- sort
- wc
- diff
- nano/vi/emacs
- awk

Текст гаралт, текст файлдай ажиллах командууд

```
$ cat *.js | wc -l  
20807
```

```
$ cat *.js > all.js
```

```
$ grep -i -E '[a-z]{6}' *.c
```

```
$ ps aux | head -10 | tail -9 | awk '{print $11}'
```

Өргөн ашиглагдах командууд

Файл системтэй ажиллах командууд

- ls
- cd
- pwd
- mkdir
- file
- cp
- mv
- rm
- ln
- chmod
- chown
- find
- locate
- du
- df
- dd
- mount
- rsync

Файл системтэй ажиллах командууд

```
$ ls -lhS
```

```
$ find src/ -name "*.c" -or -name "*.h"  
      -exec grep -H sleep {} \;
```

```
$ du -h -d 1 | sort -h
```

```
$ rsync -avzh /var/backup user@server2:/var/backup
```

Өргөн ашиглагдах командууд

Процесстой ажиллах команд

- kill / xkill / pkill / killall
- ps / pgrep / pidof
- top / htop

Процесстэй ажиллах команд

```
$ kill -9 'pidof firefox'
```

```
$ killall python3
```

Өргөн ашиглагдах командууд

Хэрэглэгчийн орчинтой ажиллах командууд

- su / sudo
- date
- alias
- uname
- uptime
- sleep

Хэрэглэгчийн орчинтой ажиллах командууд

```
$ sleep 1h && poweroff
```


Өргөн ашиглагдах командууд

Хэрэглэгч зохицуулах командууд

- useradd, userdel, usermod
- passwd
- who
- w

Хэрэглэгч зохицуулах командууд

```
$ w | grep bold
```

```
# passwd bold
```

Өргөн ашиглагдах командууд

Тусламж, тайлбар харах команд

- `man / whatis`
- `whereis`

```
man -k safety
```

Өргөн ашиглагдах командууд

Сүлжээний команд

- ip
- ping
- ifconfig/iwconfig
- traceroute
- netstat
- iptables/ufw/nft
- nc
- tcpdump

Сүлжээний команд

```
# netstat -npta
```

```
# ip addr add 10.10.10.1/24 dev enp43s0
```

Өргөн ашиглагдах командууд

Package management

- apt/dpkg
- yum/rpm
- pacman
- apk

Package management

```
# dpkg -L docker-ce
```

```
# dpkg -L docker-ce
```

```
$ apt search oauth
```

Shell-ийн төрлүүд

- bash shell (Bourne shell)
- C-shell (csh)
- KornShell (ksh)
- Z-shell (zsh) (Bourne shell)

Энгийн скрипт

Уламжлалт hello world script

————— *hello.sh* —————

```
#!/bin/bash
```

```
echo Hello wold
```

————— *hello.sh* —————

```
$ chmod +x hello.sh
```

```
$ ./hello.sh
```

Энгийн скрипт

Octal	Binary	Perms	Octal	Binary	Perms
0	000	---	4	100	r--
1	001	--x	5	101	r-x
2	010	-w-	6	110	rw-
3	011	-wx	7	111	rwX

```
$ chmod 755 hello.sh
```

Энгийн скрипт

Уламжлалт hello world script

```
----- backup.sh -----  
#!/bin/bash  
tar cvzf backup.tar.gz /home/user/  
----- backup.sh -----  
  
$ chmod +x backup.sh  
$ ./backup.sh
```

Дахин чиглүүлэх

Стандарт оролт, гаралт, алдаа

- 0 **stdin**
- 1 **stdout**
- 2 **stderr**

Стандарт гаралтыг файлууд чиглүүлэх

```
ls -l > file_list.txt
```

Стандарт алдааг файлууд чиглүүлэх

```
grep da * 2> errors.txt
```

Стандарт алдааг стандарт гаралтрууд

```
grep da * 2>&1
```

Давхар чиглүүлэг

Дараалал чухал

```
grep da * 2>&1 > output.txt      # алдаатай
grep da * > output.txt 2>&1      # зөв
```

Хувьсагчид

- Орчний хувьсагчид
- Тусгай хувьсагчид
- Хэрэглэгчийн хувьсагчид

Орчний хувьсагчид

- **printenv**
- `/etc/bash.bashrc`
- `~/.bashrc`

Тусгай хувьсагчид

\$1, \$2, \$3	positional parameters.
\$0	is the name of the shell or shell script.
\$@	is an array-like construct of all positional parameters, \$1, \$2, \$3
\$*	is the IFS expansion of all positional parameters, \$1 \$2 \$3
\$#	is the number of positional parameters.
\$_	Process number of last background command.
\$?	Exit value of last executed command.
\$	current options set for the shell.
\$\$	pid of the current shell (not subshell).
\$_	most recent parameter (or the abs path of the command to start the current shell immediately after startup).
\$IFS	is the (input) field separator.

Хэрэглэгчийн тодорхолсон хувьсагчид

Зарлах

Утга оноож хувьсагчийг зарлана. Утга оноохдоо тэнцүүгийн тэмдгийн хоёр талд ямар ч зай байж болохгүй. Жишээ нь

```
str=hello  
i=0
```

Хандах

\$<хувьсагчийн нэр>

Жишээ нь

```
$ echo $str, $i
```

Дотоод хувьсагчид

local түлхүүр үгийн тусламжтайгаар дотоод хувьсагчдыг үүсгэнэ.

```
#!/bin/bash
HELLO=Hello
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```

Нөхцөл шалгах

```
if [ expression ]; then
    statement
fi
```

```
if [ expression ]; then
    statement1
else
    statement2
fi
```

Нөхцөл шалгах

```
if [ expression1 ]; then
    statement1
elif [ expression2 ]; then
    statement2
else
    statement3
fi
```

Нөхцөл шалгах

Нөхцөл шалгах жишээ

```
#!/bin/bash

echo Тоо оруулна уу?
read N
if [ $N -le 10 ] && [ $N -ge 0 ]; then
    echo 0-10 завсар
elif [ $N -lt 0 ]; then
    echo сөрөг тоо
else
    echo 10-аас их
fi
```

-a file	True if file exists.
-b file	True if file exists and is a block special file.
-c file	True if file exists and is a character special file.
-d file	True if file exists and is a directory.
-e file	True if file exists.
-f file	True if file exists and is a regular file.
-g file	True if file exists and its set-group-id bit is set.
-h file	True if file exists and is a symbolic link.
-k file	True if file exists and its "sticky" bit is set.
-p file	True if file exists and is a named pipe (FIFO).
-r file	True if file exists and is readable.
-s file	True if file exists and has a size greater than zero.
-t fd	True if file descriptor fd is open and refers to a terminal.
-u file	True if file exists and its set-user-id bit is set.
-w file	True if file exists and is writable.
-x file	True if file exists and is executable.

Нөхцөл

-G file	True if file exists and is owned by the effective group id.
-L file	True if file exists and is a symbolic link.
-N file	True if file exists and has been modified since it was last read.
-O file	True if file exists and is owned by the effective user id.
-S file	True if file exists and is a socket.
file1 -ef file2	True if file1 and file2 refer to the same device and inode numbers.
file1 -nt file2	True if file1 is newer (according to modification date) than file2, or if file1 exists and file2 does not.
file1 -ot file2	True if file1 is older than file2, or if file2 exists and file1 does not.
-o optname	True if the shell option optname is enabled. The list of options appears in the description of the -o option to the set builtin (see The Set Builtin).
-v varname	True if the shell variable varname is set (has been assigned a value).

<code>-R varname</code>	True if the shell variable <code>varname</code> is set and is a name reference.
<code>-z string</code>	True if the length of <code>string</code> is zero.
<code>-n string</code>	True if the length of <code>string</code> is non-zero.
<code>string</code>	True if the length of <code>string</code> is non-zero.
<code>string1 == string2</code>	True if the strings are equal.
<code>string1 != string2</code>	True if the strings are not equal.
<code>string1 < string2</code>	True if <code>string1</code> sorts before <code>string2</code> lexicographically.
<code>string1 > string2</code>	True if <code>string1</code> sorts after <code>string2</code> lexicographically.
<code>arg1 OP arg2</code>	<code>OP</code> is one of <code>'-eq'</code> , <code>'-ne'</code> , <code>'-lt'</code> , <code>'-le'</code> , <code>'-gt'</code> , or <code>'-ge'</code> . These arithmetic binary operators return true if <code>arg1</code> is equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to <code>arg2</code> , respectively. <code>Arg1</code> and <code>arg2</code> may be positive or negative integers. When used with the <code>[[</code> command, <code>Arg1</code> and <code>Arg2</code> are evaluated as arithmetic expressions

Нөхцөл

Арифметик операторын урд, ард хоосон зай байх ёсгүй.

```
#!/bin/bash
if [ -e user.txt ]; then
    echo user.txt file exists
else
    echo user.txt is not found
fi
```

```
if [[ 1+3 < 5 ]]; then
    echo Yes, correct
else
    echo Nooo
fi
```

Арифметик үйлдлүүд

((хаалт эсвэл **let** командыг ашиглан арифметик үйлдлийг хийнэ.

<code>id ++ id --</code>	variable post-increment and post-decrement
<code>++ id -- id</code>	variable pre-increment and pre-decrement
<code>- +</code>	unary minus and plus
<code>! ~</code>	logical and bitwise negation
<code>**</code>	exponentiation
<code>*/%</code>	multiplication, division, remainder
<code>+-</code>	addition, subtraction
<code><< >></code>	left and right bitwise shifts
<code><= >= < ></code>	comparison
<code>== !=</code>	equality and inequality
<code>&</code>	bitwise AND
<code>^</code>	bitwise exclusive OR
<code> </code>	bitwise OR
<code>&&</code>	logical AND
<code> </code>	logical OR

Арифметик үйлдлүүд

((хаалт эсвэл **let** командыг ашиглан арифметик үйлдлийг хийнэ.

expr1, expr2

expr ? expr : expr

= * = / = % = + = - = << = >> = & = ^ = | =

Арифметик үйлдлүүд

```
#!/bin/bash  
i=0  
((i++))  
let "i *= 3"  
echo $i
```

Хариу: 3

Давталт

```
#!/bin/bash
for ((i = 0; i < 100; i++)); do
    echo $i
done
```

Дэд командаар гаралтаар гүйх

```
#!/bin/bash
for i in $( ls ); do
    echo item: $i
done
```

Давталт

while давталт

```
i=0
while [ $i -lt 100 ]; do
    if [[ $i%2 -eq 0 ]]; then
        echo $i
    fi
    ((i++))
done
```

Функц

Аргумент авдаг функц

Bash функцаас утга буцаах боломжгүй байдаг.

```
#!/bin/bash
function is_even {
    if [[ $1%2 -eq 0 ]]; then
        ret=1
    else
        ret=0
    fi
}

for ((i = 1; i <= 100; i++)); do
    is_even $i
    if [ $ret -eq 1 ]; then
        echo $i
    fi
done
```

Хүснэгт

Зарлах

```
name[index]=value  
declare -a name  
name=(value1 value2 ...)
```

Элементэд хандах

```
${name[index]}
```


Хүснэгт

Гараас оруулсан утгыг хүснэгтэд хадгалах

```
#!/bin/bash
declare -a a
for ((i = 0; i < 5; i++)) do
    read a[$i]
done

for ((i = 0; i < 5; i++)) do
    echo ${a[$i]}
done
```

Цэснээс сонгох

```
OPTIONS="Yes No"
select opt in $OPTIONS; do
    if [ "$opt" = "Yes" ]; then
        echo "you selected yes"
        exit 0
    elif [ "$opt" = "No" ]; then
        echo "you selected no"
        exit 0
    else
        echo "it's something else"
        exit 1
    fi
done
```

Debug хийх

Алдаагаа олж чадахгүй бол

```
#!/bin/bash -x
```

Эх сурвалж

- 1 <http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>
- 2 <https://www.gnu.org/software/bash/manual/>